2007

Compact Spectral Bases for Value Function Approximation Using Kronecker Factorization

Jeff Johns
University of Massachusetts Amherst

Compact Spectral Bases for Value Function Approximation Using Kronecker Factorization*

Jeff Johns and **Sridhar Mahadevan** and **Chang Wang**
Computer Science Department
University of Massachusetts Amherst
Amherst, Massachusetts 01003
{johns, mahadeva, chwang}@cs.umass.edu

Abstract

A new spectral approach to value function approximation has recently been proposed to automatically construct basis functions from samples. Global basis functions called proto-value functions are generated by diagonalizing a diffusion operator, such as a reversible random walk or the Laplacian, on a graph formed from connecting nearby samples. This paper addresses the challenge of scaling this approach to large domains. We propose using Kronecker factorization coupled with the Metropolis-Hastings algorithm to decompose reversible transition matrices. The result is that the basis functions can be computed on much smaller matrices and combined to form the overall bases. We demonstrate that in several continuous Markov decision processes, compact basis functions can be constructed without significant loss in performance. In one domain, basis functions were compressed by a factor of 36. A theoretical analysis relates the quality of the approximation to the spectral gap. Our approach generalizes to other basis constructions as well.

Introduction

Value function approximation is a critical step in solving large Markov decision processes (MDPs) (Bertsekas & Tsitsiklis 1996). A well-studied approach is to approximate the (action) value function $V^\pi(s)$ ($Q^\pi(s,a)$) of a policy π by least-squares projection onto the linear subspace spanned by a set of basis functions forming the columns of a matrix Φ:

$$V^\pi = \Phi w^\pi \qquad\qquad Q^\pi = \Phi w^\pi$$

For approximating action-value functions, the basis function matrix Φ is defined over state-action pairs (s,a), whereas for approximating value functions, the matrix is defined over states. The choice of bases is an important decision for value function approximation. The majority of past work has typically assumed basis functions can be hand engineered. Some popular choices include tiling, polynomials, and radial basis functions (Sutton & Barto 1998).

*This research was supported in part by the National Science Foundation under grant NSF IIS-0534999.

Since manual construction of bases can be a difficult trial-and-error process, it is natural to devise an algorithmic solution to the problem. Several promising approaches have recently been proposed that suggest feature discovery in MDPs can be automated. This paper builds on one of those approaches: the Representation Policy Iteration (RPI) framework (Mahadevan 2005). Basis functions in RPI are derived from a graph formed by connecting nearby states in the MDP. The basis functions are eigenvectors of a diffusion operator (e.g. the random walk operator or the graph Laplacian (Chung 1997)). This technique yields geometrically customized, global basis functions that reflect topological singularities such as bottlenecks and walls.

Spectral basis functions are not compact since they span the set of samples used to construct the graph. This raises a computational issue of whether this approach (and related approaches such as Krylov bases (Petrik 2007)) scale to large MDPs. In this paper, we explore a technique for making spectral bases compact. We show how a matrix A (representing the random walk operator on an arbitrary, weighted undirected graph) can be factorized into two smaller stochastic matrices B and C such that the Kronecker product $B \otimes C \approx A$. This procedure can be called recursively to further shrink the size of B and/or C. The Metropolis-Hastings algorithm is used to make B and C reversible, which ensures their eigendecompositions contain all real values. The result is the basis functions can be calculated from B and C rather than the original matrix A. This is a gain in terms of both speed and memory. We demonstrate this technique using three standard benchmark tasks: inverted pendulum, mountain car, and Acrobot. The basis functions in the Acrobot domain are compressed by a factor of 36. There is little loss in performance by using the compact basis functions to approximate the value function. We also provide a theoretical analysis explaining the effectiveness of the Kronecker factorization.

Algorithmic Framework

Figure 1 presents a generic algorithmic framework for learning representation and control in MDPs based on (Mahadevan *et al.* 2006), which comprises of three phases: an initial sample collection phase, a basis construction phase, and a parameter estimation phase. As shown in the figure, each phase of the overall algorithm includes optional *basis spar-*

Sample Collection Phase:

1. Generate a dataset $\mathcal{D}$ of "state-action-reward-nextstate" transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ using a series of random walks in the environment (terminated by reaching an absorbing state or some fixed maximum length).
2. **Sparsification Step I:** Subsample a set of transitions $\mathcal{D}_s$ from $\mathcal{D}$ by some method (e.g. randomly or greedily).

Representation Learning Phase

3. Construct a *diffusion model* from $\mathcal{D}_s$ consisting of an undirected graph $G = (V, E, W)$, with edge set E and weight matrix W. Each vertex $v \in V$ corresponds to a visited state. Edges are inserted between a pair of vertices x_i and x_j if x_j is among the k nearest neighbors of x_i, with weight $W(i,j) = \alpha(i)e^{-\frac{d(x_i,x_j)}{\sigma}}$, where $\sigma > 0$ is a parameter, $d(x_i, x_j)$ is an appropriate distance metric (e.g. Euclidean $\|x_i - x_j\|^2_{\mathcal{R}^m}$), and α a specified weight function. Symmetrize $W_S = (W + W^T)/2$. Then $A = D^{-1}W_S$ is the random walk operator, where D is a diagonal matrix of row sums of W_S.
4. **Sparsification Step II:** Reorder matrix A to cluster similar states (e.g. using a graph partitioning program). Compute stochastic matrices B (size $r_B \times c_B$) and C (size $r_C \times c_C$) such that $B \otimes C \approx A$ (time $O(r_B^2 r_C^2)$). Use the Metropolis-Hastings algorithm to convert these matrices into *reversible* stochastic matrices B_R and C_R (time $O(r_B^2 + r_C^2)$). Optional: call this step recursively.
5. Calculate the eigenvalues λ_i (μ_j) and eigenvectors x_i (y_j) of B_R (C_R). This takes time $O(r_B^3 + r_C^3)$ compared to computing A's eigenvectors in time $O(r_B^3 r_C^3)$. The basis matrix Φ *could* be explicitly calculated by selecting the "smoothest" ℓ eigenvectors of $x_i \otimes y_j$ (corresponding to the largest eigenvalues $\lambda_i \mu_j$) as columns of Φ. However, to take advantage of the compact nature of the Kronecker product, Φ should not be explicitly calculated; state embeddings can be computed when needed by the parameter estimation algorithm. Φ stores $(r_B r_C \ell)$ values in memory whereas the eigenvectors of B_R and C_R only store $(r_B \cdot \min(r_B, \ell) + r_C \cdot \min(r_C, \ell))$ values.

Control Learning Phase

6. Use a parameter estimation method such as LSPI (Lagoudakis & Parr 2003) or Q-learning (Sutton & Barto 1998) to find the best policy π in the linear span of Φ, where the action value functions $Q^\pi(s,a)$ is approximated as $Q^\pi \approx \Phi w^\pi$.
7. **Sparsification Step III:** Eliminate basis functions whose coefficients in w^π fall below a threshold.

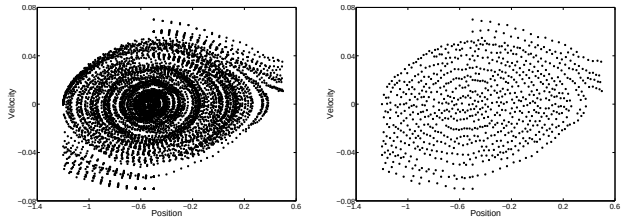
Figure 1: The RPI framework for learning representation and control in MDPs.

sification steps. The main contribution of this paper is the second sparsification step. The computational complexity of steps 4 and 5 are shown in the figure to highlight the savings.

One of the main benefits of the Kronecker factorization is that the basis matrix Φ does not need to be explicitly calculated (step 5 in Figure 1). The matrix is stored in a *compact* form as the eigenvectors of matrices B_R and C_R. Parameter estimation algorithms, such as LSPI, require state (or state-action) embeddings that correspond to a single row in the matrix Φ. This row can be computed by indexing into the appropriate eigenvectors of B_R and C_R. Thus, memory requirements are reduced and can be further minimized by recursively factorizing B_R and/or C_R.

Sparsifying Bases by Sampling

Spectral bases are amenable to sparsification methods investigated in the kernel methods literature including low-rank approximation techniques as well as the Nyström interpolation method (Drineas & Mahoney 2005) for extrapolating eigenvectors on sampled states to novel states. We have found subsampling the states using a greedy algorithm greatly reduces the number of samples while still capturing the structure of the data manifold. The greedy algorithm is simple: starting with the null set, add samples to the subset that are **not** within a specified distance to any sample currently in the subset. A maximal subset is returned when no more samples can be added. In the experiments reported in this paper, where states are continuous vectors $\in \mathcal{R}^m$, typically only 10% of the transitions in the random walk dataset $\mathcal{D}$ are necessary to learn an adequate set of basis functions. For example, in the mountain car task, 700 samples are sufficient to form the basis functions, whereas 7,000 samples are usually needed to learn a stable policy. Figure 2 shows the results of the greedy subsampling algorithm on data from this domain.



(a) 7,459 samples (b) 846 subsamples

Figure 2: Greedy subsampling in the mountain car task.

Kronecker Product

The Kronecker product of a $r_B \times c_B$ matrix B and a $r_C \times c_C$ matrix C is equal to a matrix A of size $(r_B r_C) \times (c_B c_C)$ with block $A_{i,j} = B(i,j)C$. Thus, every (i,j) block of A is equal to the matrix C multiplied by the scalar $B(i,j)$. The equation is written $A = B \otimes C$. The Kronecker product can be used to streamline many computations in numerical linear algebra, signal processing, and graph theory.

We focus in this paper on the case where B and C correspond to stochastic matrices associated with weighted, undirected graphs $G_B = (V_B, E_B, W_B)$ and $G_C = (V_C, E_C, W_C)$ respectively. The graphs can be represented as weight matrices W_B and W_C. We assume strictly positive edge weights. Matrix B is then formed by dividing each row of W_B by the row sum (similarly for C). B and C are stochastic matrices representing random walks over their respective graphs. The eigenvalues and eigenvectors of B and C completely determine the eigenvalues and eigenvectors of $B \otimes C$.

Theorem 1: Let B have eigenvectors x_i and eigenvalues λ_i for $1 \leq i \leq r_B$. Let C have eigenvectors y_j and eigenvalues μ_j for $1 \leq j \leq r_C$. Then matrix $B \otimes C$ has eigenvectors $x_i \otimes y_j$ and eigenvalues $\lambda_i \mu_j$.

Proof: Consider $(B \otimes C)(x_i \otimes y_j)$ evaluated at vertex (v, w) where $v \in V_B$ and $w \in V_C$.

$$
\begin{aligned}
&(B \otimes C)(x_i \otimes y_j)(v, w) \\
&= \sum_{(v,v_2) \in E_B} \sum_{(w,w_2) \in E_C} B(v, v_2) C(w, w_2) x_i(v_2) y_j(w_2) \\
&= \sum_{(v,v_2) \in E_B} B(v, v_2) x_i(v_2) \sum_{(w,w_2) \in E_C} C(w, w_2) y_j(w_2) \\
&= (\lambda_i x_i(v))\,(\mu_j y_j(w)) \;=\; (\lambda_i \mu_j)\,(x_i(v) y_j(w))
\end{aligned}
$$

We adapted this theorem from a more general version (Bellman 1970) that does not place constraints on the two matrices. Note this theorem also holds if B and C are normalized Laplacian matrices (Chung 1997), but it does not hold for the combinatorial Laplacian. The Kronecker product is an important tool because the eigendecomposition of $A = B \otimes C$ can be accomplished by solving the smaller problems on B and C individually. The computational complexity is reduced from $O(r_B^3 r_C^3)$ to $O(r_B^3 + r_C^3)$.

Kronecker Product Approximation

Given the computational benefits of the Kronecker factorization, it is natural to consider the problem of finding matrices B and C to approximate a matrix A. Pitsianis (1997) studied this problem for *arbitrary* matrices. Specifically, given a matrix A, the problem is to minimize the function

$$f_A(B, C) = \|A - B \otimes C\|_F, \tag{1}$$

where $\| \cdot \|_F$ is the Frobenius norm. By reorganizing the rows and columns of A, the function f_A can be rewritten

$$f_A(B, C) = \|\tilde{A} - \text{vec}(B)\text{vec}(C)^T\|_F \tag{2}$$

where the $\text{vec}(\cdot)$ operator takes a matrix and returns a vector by stacking the columns in order. The matrix $\tilde{A}$ is defined

$$\tilde{A} = \begin{bmatrix} \text{vec}(A_{1,1})^T \\ \vdots \\ \text{vec}(A_{r_B,1})^T \\ \vdots \\ \text{vec}(A_{1,c_B})^T \\ \vdots \\ \text{vec}(A_{r_B,c_B})^T \end{bmatrix} \in \mathcal{R}^{(r_B c_B) \times (r_C c_C)}. \tag{3}$$

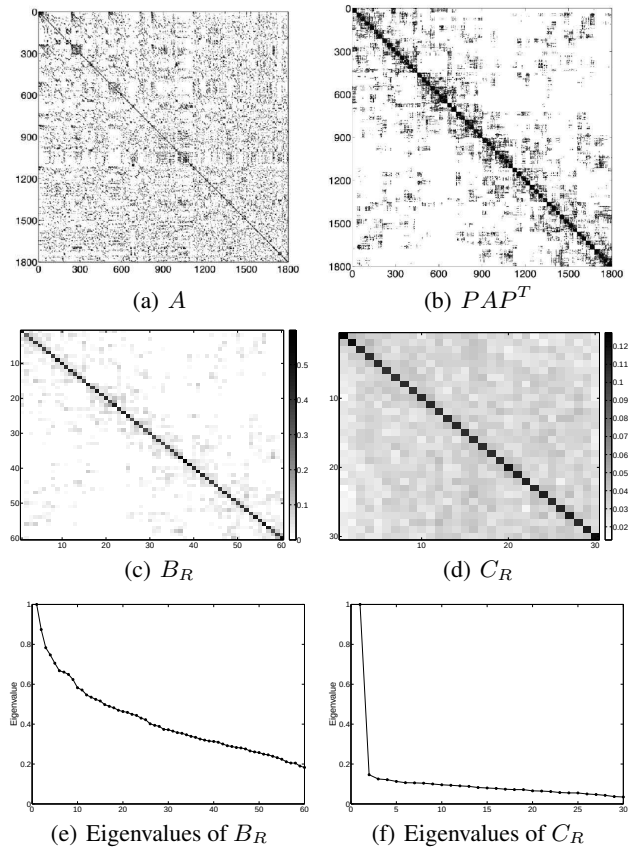
Equation 2 shows the Kronecker product approximation problem is equivalent to a rank-one matrix problem. The solution to a rank-one matrix problem can be computed from the singular value decomposition (SVD) of $\tilde{A} = U \Sigma V^T$ (Golub & Van Loan 1996). The minimizing values are $\text{vec}(B) = \sqrt{\sigma_1} u_1$ and $\text{vec}(C) = \sqrt{\sigma_1} v_1$ where u_1 and v_1 are the first columns of U and V and σ_1 is the largest singular value of $\tilde{A}$. This is done in time $O(r_B^2 r_C^2)$ since only the first singular value and singular vectors of the SVD are required.

Pitsianis (1997) extended this idea to constrained optimization problems where the matrices B and C are required to have certain properties: symmetry, orthogonality, and stochasticity are examples. In this paper, we used the `kpa_markov` algorithm which finds stochastic matrices B and C that approximate a stochastic matrix A given as input. There are equality (row sums must equal one) and inequality (all values must be non-negative) constraints for this problem. The `kpa_markov` algorithm substitutes the equality constraints into the problem formulation and ignores the inequality constraints. One iteration of the algorithm proceeds by fixing C and updating B based on the derivative of $\|A - B \otimes C\|_F$; then matrix B is held constant and C is updated. Convergence is based on the change in the Frobenius norm. The algorithm used at most 6 iterations in our experiments. If the algorithm returned negative values, those entries were replaced with zeros and the rows were rescaled to sum to 1. More sophisticated algorithms (e.g. active set method) could be used to directly account for the inequality constraints if necessary.

The Kronecker product has simple semantics when the matrices are stochastic. Matrix A is compacted into r_B clusters, each of size r_C. Matrix B contains transitions between clusters while matrix C contains transitions within a cluster. For the block structure of the Kronecker product to be most effective, similar states must be clustered. This can be achieved by reordering matrix A via PAP^T where P is a permutation matrix. The problem of finding the optimal P to minimize $\|PAP^T - B \otimes C\|_F$ is NP-hard. However, there are several options for reordering matrices including graph partitioning and approximate minimum degree ordering. We used the graph partitioning program METIS (Karypis & Kumar 1999) to determine P. METIS combines several heuristics for generating partitions, optimizing the balance of a partition versus the number of edges going across partitions. The algorithm first coarsens the graph, then partitions the smaller graph, and finally uncoarsens and refines the partitions. METIS is a highly optimized program that partitions graphs with 10^6 vertices in a few seconds. Figure 3(a) shows an adjacency plot of a matrix A corresponding to a graph connecting 1800 sample states from the Acrobot domain. Figure 3(b) is the same matrix but reordered according to METIS with 60 partitions. The reordered matrix is in a block structure more easily represented by the Kronecker decomposition.

The stochastic matrices B and C are not necessarily reversible. As such, their eigenvalues can be complex. To ensure all real values, we used the Metropolis-Hastings algorithm to convert B and C into reversible stochastic matrices B_R and C_R. The algorithm is described below where π is a stationary probability distribution.

$$
B_R(i,j) = \begin{cases}
B(i,j) \min\left(1, \dfrac{\pi(j) B(j,i)}{\pi(i) B(i,j)}\right) & \text{if } i \neq j \\
B(i,j) + \displaystyle\sum_k B(i,k) & \text{if } i = j \\
\quad \times \left(1 - \min\left(1, \dfrac{\pi(k) B(k,i)}{\pi(i) B(i,k)}\right)\right)
\end{cases}
$$



(a) A (b) PAP^T

(c) B_R (d) C_R

(e) Eigenvalues of B_R (f) Eigenvalues of C_R

Figure 3: (a) Adjacency plot of an 1800×1800 matrix from the Acrobot domain, (b) Matrix reordered using METIS, (c) 60×60 matrix B_R, (d) 30×30 matrix C_R, (e) Spectrum of B_R, and (f) Spectrum of C_R.

This transformation was proven (Billera & Diaconis 2001) to minimize the distance in an L^1 metric between the original matrix B and the space of reversible stochastic matrices with stationary distribution π. We used the power method (Golub & Van Loan 1996) to determine the stationary distributions of B and C. Note these stationary distributions were unique in our experiments because B and C were both aperiodic and irreducible although the `kpa_markov` algorithm does not specifically maintain these properties. The Frobenius norm between B and B_R (and between C and C_R) was small in our experiments. Figures 3(c) and 3(d) show grayscale images of the reversible stochastic matrices B_R and C_R that were computed by this algorithm to approximate the matrix in Figure 3(b). As these figures suggest, the Kronecker factorization is performing a type of state aggregation. The matrix B_R has the same structure as PAP^T, whereas C_R is close to a uniform block matrix except with more weight along the diagonal. The eigenvalues of B_R and C_R are displayed in Figures 3(e) and 3(f). The fact that C_R is close to a block matrix can be seen in the large gap between the first and second eigenvalues.

It is more economical to store the eigenvectors of B_R and C_R than those of A. We used 90 eigenvectors for our experiments in the Acrobot domain; thus, the eigenvectors of matrix A in Figure 3(a) consist of 162,000 values (1800×90). There are 3,600 values (60×60) for B_R and 900 values (30×30) for C_R, yielding a compression ratio of 36.

There is an added benefit of computing the stationary distributions. The eigendecomposition of B_R (and C_R) is less robust because the matrix is unsymmetric. However, B_R is *similar* to a symmetric matrix $B_{R,S}$ by the equation $B_{R,S} = \Pi^{0.5} B_R \Pi^{-0.5}$ where Π is a diagonal matrix with elements π. Matrices B_R and $B_{R,S}$ have identical eigenvalues and the eigenvectors of B_R can be computed by multiplying $\Pi^{-0.5}$ by the eigenvectors of $B_{R,S}$. Therefore, the decomposition should always be done on $B_{R,S}$.

Theoretical Analysis

This analysis attempts to shed some light on when $B \otimes C$ is useful for approximating A. More specifically, we are concerned with whether the space spanned by the top m eigenvectors of $B \otimes C$ is "close" to the space spanned by the top m eigenvectors of A. Perturbation theory can be used to address this question because the random walk operator A is self-adjoint (with respect to the *invariant distribution* of the random walk) on an inner product space; therefore, theoretical results concerning A's spectrum apply. We assume matrices B and C are computed according to the constrained Kronecker product approximation algorithm discussed in the previous section. The following notation is used in the theorem and proof:

- $E = A - B \otimes C$
- X is a matrix whose columns are the top m eigenvectors of A
- α_1 is the set of top m eigenvalues of A
- α_2 includes all eigenvalues of A except those in α_1
- d is the eigengap between α_1 and α_2, i.e. $d = \min_{\lambda_i \in \alpha_1, \lambda_j \in \alpha_2} |\lambda_i - \lambda_j|$
- Y is a matrix whose columns are the top m eigenvectors of $B \otimes C$
- $\tilde{\alpha_1}$ is the set of top m eigenvalues of $B \otimes C$
- $\tilde{\alpha_2}$ includes all eigenvalues of $B \otimes C$ except those in $\tilde{\alpha_1}$
- $\tilde{d}$ is the eigengap between α_1 and $\tilde{\alpha_2}$
- $\mathcal{X}$ is the subspace spanned by X
- $\mathcal{Y}$ is the subspace spanned by Y
- P is the orthogonal projection onto $\mathcal{X}$
- Q is the orthogonal projection onto $\mathcal{Y}$

Theorem 2: Assuming B and C are defined as above based on the SVD of $\tilde{A}$ and if $\|E\| \leq 2\varepsilon d/(\pi + 2\varepsilon)$, then the distance between the space spanned by top m eigenvectors of A and the space spanned by the top m eigenvectors of $B \otimes C$ is at most ε.

Proof: The Kronecker factorization uses the top m eigenvectors of $B \otimes C$ to approximate the top m eigenvectors of A (e.g. use Y to approximate X). The difference between $\mathcal{X}$ and $\mathcal{Y}$ is defined $\|Q - P\|$. [$\mathbf{S}_1$]

It can be shown that if A and E are bounded self-adjoint operators on a separable Hilbert space, then the spectrum of A+E is in the closed $\|E\|$-neighborhood of the spectrum of A (Kostrykin, Makarov, & Motovilov 2003). The authors also prove the inequality $\|Q^{\perp}P\| \leq \pi\|E\|/2\tilde{d}$. [$\mathbf{S}_2$]

Matrix A has an isolated part α_1 of the spectrum separated from its remainder α_2 by gap d. To guarantee A+E also has separated spectral components, we need to assume $\|E\| < d/2$. Making this assumption, [$\mathbf{S}_2$] can be rewritten $\|Q^{\perp}P\| \leq \pi\|E\|/2(d - \|E\|)$. [$\mathbf{S}_3$]

Interchanging the roles of α_1 and α_2, we have the analogous inequality: $\|QP^{\perp}\| \leq \pi\|E\|/2(d - \|E\|)$. [$\mathbf{S}_4$] Since $\|Q - P\| = \max\{\|Q^{\perp}P\|, \|QP^{\perp}\|\}$ [$\mathbf{S}_5$], the overall inequality can be written $\|Q-P\| \leq \pi\|E\|/2(d-\|E\|)$. [$\mathbf{S}_6$]

Step [$\mathbf{S}_6$] implies that if $\|E\| \leq 2\varepsilon d/(\pi + 2\varepsilon)$, then $\|Q - P\| \leq \varepsilon$. [$\mathbf{S}_7$]

The two important factors involved in this theorem are $\|E\|$ and the eigengap of A. If $\|E\|$ is small, then the space spanned by the top m eigenvectors of $B \otimes C$ approximates the space spanned by the top m eigenvectors of A well. Also, for a given value of $\|E\|$, the larger the eigengap the better the approximation.

Experimental Results

Testbeds

Inverted Pendulum: The inverted pendulum problem requires balancing a pendulum of unknown mass and length by applying force to the cart to which the pendulum is attached. We used the implementation described in (Lagoudakis & Parr 2003). The state space is defined by two variables: θ, the vertical angle of the pendulum, and $\dot{\theta}$, the angular velocity of the pendulum. The three actions are applying a force of -50, 0, or 50 Newtons. Uniform noise from -10 and 10 is added to the chosen action. State transitions are described by the following nonlinear equation

$$\ddot{\theta} = \frac{g\sin(\theta) - \alpha ml\dot{\theta}^2\sin(2\theta)/2 - \alpha\cos(\theta)u}{4l/3 - \alpha ml\cos^2(\theta)},$$

where u is the noisy control signal, $g = 9.8m/s^2$ is gravity, $m = 2.0$ kg is the mass of the pendulum, $M = 8.0$ kg is the mass of the cart, $l = 0.5$ m is the length of the pendulum, and $\alpha = 1/(m + M)$. The simulation time step is set to 0.1 seconds. The agent is given a reward of 0 as long as the absolute value of the angle of the pendulum does not exceed $\pi/2$, otherwise the episode ends with a reward of -1. The discount factor was set to 0.9. The maximum number of episodes the pendulum was allowed to balance was 3,000 steps.

Mountain Car: The goal of the mountain car task is to get a simulated car to the top of a hill as quickly as possible. The car does not have enough power to get there immediately, and so must oscillate on the hill to build up the necessary momentum. This is a minimum time problem, and thus the reward is -1 per step. The state space includes the position and velocity of the car. There are three actions: full throttle forward (+1), full throttle reverse (-1), and zero throttle (0). The position, x_t, and velocity, $\dot{x}_t$, are updated by

$$x_{t+1} = \text{bound}[x_t + \dot{x}_{t+1}]$$
$$\dot{x}_{t+1} = \text{bound}[\dot{x}_t + 0.001a_t + -0.0025\cos(3x_t)],$$

where the bound operation enforces $-1.2 \leq x_{t+1} \leq 0.6$ and $-0.07 \leq \dot{x}_{t+1} \leq 0.07$. The episode ends when the car successfully reaches the top, defined as position $x_t \geq 0.5$. The discount factor was 0.99 and the maximum number of test steps was 500.

Acrobot: The Acrobot task (Sutton & Barto 1998) is a two-link under-actuated robot that is an idealized model of a gymnast swinging on a highbar. The task has four continuous state variables: two joint positions and two joint velocities. The only action available is a torque on the second joint, discretized to one of three values (positive, negative, and none). The reward is -1 for all transitions leading up to the goal state. The detailed equations of motion are given in (Sutton & Barto 1998). The discount factor was set to 1 and we allowed a maximum of 600 steps before terminating without success in our experiments.

Experiments

The experiments followed the framework outlined in Figure 1. The first sparsification step was done using the greedy subsampling procedure. Graphs were then built by connecting each subsampled state to its k nearest neighbors and edge weights were assigned using a *weighted* Euclidean distance metric. A weighted Euclidean distance metric was used as opposed to an unweighted metric to make the state space dimensions have more similar ranges. These parameters are given in the first three rows of Table 1. There is one important exception for graph construction in Acrobot. The joint angles θ_1 and θ_2 range from 0 to 2π; therefore, arc length is the appropriate distance metric to ensure values slightly greater than 0 are "close" to values slightly less than 2π. However, the fast nearest neighbor code that we used to generate graphs required a Euclidean distance metric. To approximate arc length using Euclidean distance, angle θ_i was mapped to a tuple $[\sin(\theta_i), \cos(\theta_i)]$ for $i = \{1, 2\}$. This approximation works very well if two angles are similar (e.g. within 30 degrees of each other) and becomes worse as the angles are further apart. Next, matrices A, B_R, and C_R were computed according to steps 4 and 5 in Figure 1. By fixing the size of C_R, the size of B_R is automatically determined ($|A| = |B_R| \cdot |C_R|$). The last four rows of Table 1 show the sizes of B_R and C_R, the number of eigenvectors used, and the compression ratios achieved by storing the compact basis functions. The LSPI algorithm was used to learn a policy.

The goal of our experiments was to compare the effectiveness of the basis functions derived from matrix A (e.g. the eigenvectors of the random walk operator) with the basis functions derived from matrices B_R and C_R. We ran 30 separate runs for each domain varying the number of episodes. The learned policies were evaluated until the goal was reached or the maximum number of steps exceeded. The median test results over the 30 runs are plotted in Figure 4. Performance was consistent across the three domains: the policy determined by LSPI achieved similar performance,

	Inverted Pendulum	Mountain Car	Acrobot
k	25	25	25
σ	1.5	0.5	3.0
Weight	[3, 1]	[1, 3]	[1, 1, 0.5, 0.3]
	$[\theta, \dot{\theta}]$	$[x_t, \dot{x_t}]$	$[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$
Eigenvectors	50	50	90
Size C_R	10	10	30
Size B_R	≈ 35	≈ 100	≈ 60
Compression	≈ 13.2	≈ 12.2	≈ 36.0

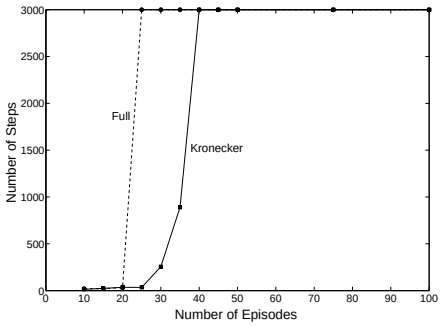
Table 1: Parameters for the experiments.

albeit more slowly, with the $B_R \otimes C_R$ basis functions than the basis functions from A. The variance from run to run is relatively high (not shown to keep the plots legible), indicating the difference between the two curves is not significant. These results show the basis functions can be made compact without hurting performance.

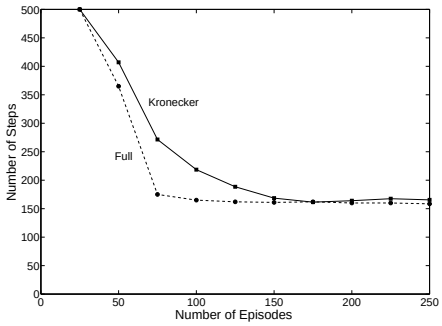
Future Work

Kronecker factorization was used to speed up construction of spectral basis functions and to store them more compactly. Experiments in three continuous MDPs demonstrate how these compact basis functions still provide a useful subspace for value function approximation. The formula for the Kronecker product suggests factorization is performing a type of state aggregation. We plan to explore this connection more formally in the future. The relationship between the size of C_R, which was determined empirically in this work, and the other parameters will be explored. We also plan to test this technique in larger domains.

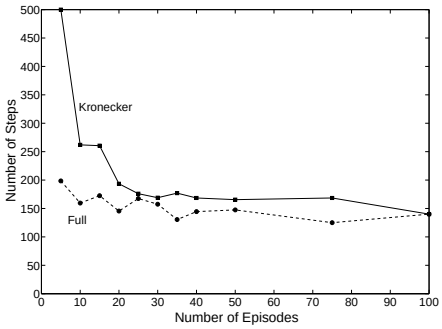
Ongoing work includes experiments with a multilevel recursive Kronecker factorization. Preliminary results have been favorable in the inverted pendulum domain using a two level factorization.



(a) Inverted Pendulum



(b) Mountain Car



(c) Acrobot

Figure 4: Median performance over the 30 runs using the RPI algorithm. The basis functions are either derived from matrix A (Full) or from matrices B_R and C_R (Kronecker).

References

Bellman, R. 1970. *Introduction to Matrix Analysis*. New York, NY: McGraw-Hill Education, 2nd edition.

Bertsekas, D., and Tsitsiklis, J. 1996. *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific.

Billera, L., and Diaconis, P. 2001. A geometric interpretation of the Metropolis-Hasting algorithm. *Statist. Science* 16:335–339.

Chung, F. 1997. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. Providence, RI: American Mathematical Society.

Drineas, P., and Mahoney, M. 2005. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research* 6:2153–2175.

Golub, G., and Van Loan, C. 1996. *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 3rd edition.

Karypis, G., and Kumar, V. 1999. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* 20(1):359–392.

Kostrykin, V.; Makarov, K. A.; and Motovilov, A. K. 2003. On a subspace perturbation problem. In *Proc. of the American Mathematical Society*, volume 131, 1038–1044.

Lagoudakis, M., and Parr, R. 2003. Least-Squares Policy Iteration. *Journal of Machine Learning Research* 4:1107–1149.

Mahadevan, S.; Maggioni, M.; Ferguson, K.; and Osentoski, S. 2006. Learning representation and control in continuous Markov decision processes. In *Proc. of the 21st National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Mahadevan, S. 2005. Representation Policy Iteration. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 372–379. Arlington, VA: AUAI Press.

Petrik, M. 2007. An analysis of Laplacian methods for value function approximation in MDPs. In *Proc. of the 20th International Joint Conference on Artificial Intelligence*, 2574–2579.

Pitsianis, N. 1997. *The Kronecker Product in Approximation and Fast Transform Generation*. Ph.D. Dissertation, Department of Computer Science, Cornell University, Ithaca, NY.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning*. Cambridge, MA: MIT Press.